Fortran program for root estimation using Newton-Raphson method

Written by Evgenii Neumerzhitckii

Aug 9, 2019

Abstract

We write a Fortran program that finds a root of equation $\cos(x) - x = 0$ using Newton-Raphson method. We find the root to be 0.73909. We investigate the dependence of the failure rate of the program on the initial value of x and the maximum number of allowed iterations.

Overview

We created a Fortran program that estimates a single root of equation

$$\cos(x) - x = 0 \tag{1}$$

using Newton-Raphson method. The method gives recurrence relation

$$x_{x+1} = x_n - f(x_n)/f'(x_n), \tag{2}$$

where

$$f(x_n) = \cos(x_n) - x_n,$$

and x_n are the x values, and $n = 0, 1, 2, \ldots, N_{max}$ is the iteration number with N_{max} being the maximum number of iterations.

We begin the calculations by choosing a starting x_0 and then use Equation 2 to calculate x_1. Then we use x_1 to calculate x_2. This calculation is repeated until the absolute difference between two subsequent x values is smaller than a chosen tolerance number ϵ:

$$|x_{n+1} - x_n| < \epsilon.$$

The calculations are stopped and the program is terminated with an error if the number of iterations exceeds a chosen maximum number of iterations. The program is also terminated if devision by zero or an overflow is detected as a result of calculating x_{x+1} from Equation 2.

Instructions for compiling and running the program are located in the README.md file that comes with the source code.

Root finding function

We implemented a function `approximate_root` with interface shown in Listing 1.

Listing 1: Definition of a function for approximating a root of equation that is passed as input parameter (`newton_raphson.f90`).

```
function approximate_root(x_start, func, derivative, tolerance, &
                          max_iterations, success) result(result)

...
end function
```

When calling `approximate_root` function, we supply a function that calculates

$$f(x) = \cos(x) - x,$$

as well as its derivative. This implementation allows to make `approximate_root` function general and reuse it for calculating roots of other functions.

Choosing initial x value

We can estimate approximate location of the root of Equation 1 by evaluating $f(x) = \cos(x) - 1$ until we find two x values x_a and x_b for which f has values of opposite signs. Intermediate value theorem guarantees that $f(x) = 0$ for some $x \in [x_a, x_b]$, since f is continuous. Therefore, we can chose initial x value x_0 to be somewhere between x_a and x_b because it will be close to a root. For example, we can chose x_0 to be between 0 and π, since $f(0) = 1$ and $f(\pi/2) = -\pi/2$ have opposite signs.

Failure ratio dependence on x_0

We wanted to know how the failure ratio of our program depends on the initial value x_0. We ran the program using multiple values of x_0 with the interval $[-10, 10]$ using tolerance $\epsilon = 1 \times 10^{-5}$ and calculated the failure ratio

$$\text{failure ratio} = \frac{\text{number of failures}}{\text{total number of runs}}.$$

We have repeated this method using different number of N_{max}:

$$N_{max} \in \{1, 5, 10, 20, 50, 100, 1000, 10\,000, 100\,000, 1\,000\,000\}.$$

For $N_{max} = 1$ all calculations failed (failure ratio of 1.0). For $N_{max} = 5$ some of the calculations succeeded, as shown in Figure 1. We can see that even five iterations were enough to reach nearly zero failure ratio in the interval between 0 and 2.5.

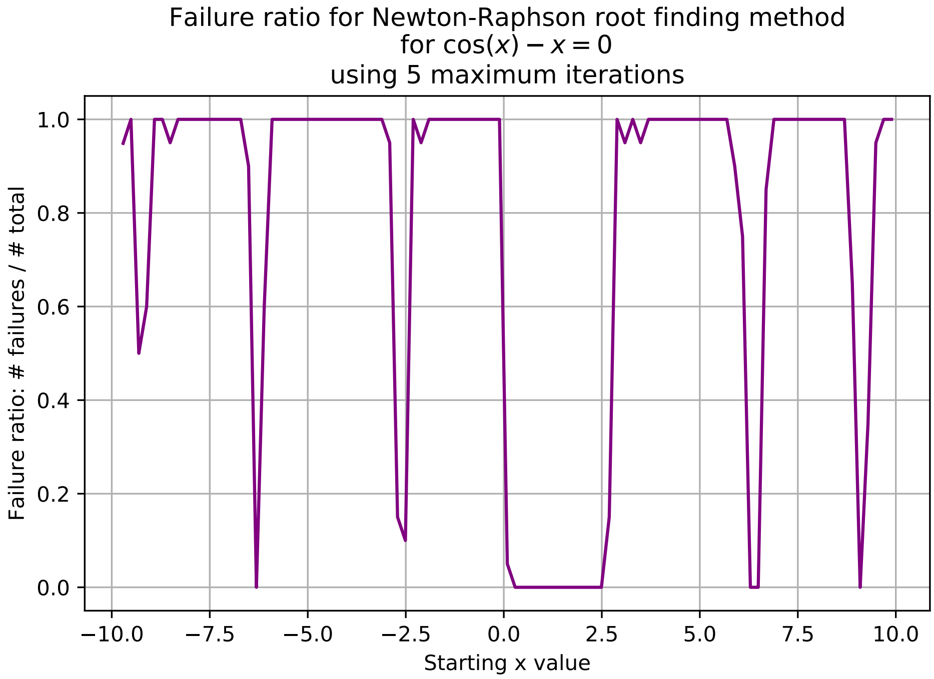

Figure 1: Failure ratio of Newton-Raphson root finding method for five maximum number of iterations. Failure ratio is close to zero in the interval $[0, 2.5]$.

Increasing the number of iterations to 50 has decreased the failure ratio for most values of x_0 to 0.6 and lower (Figure 2). However, we can still see that all calculation failed for x_0 between about -1 and 0. Another peak in failure rates is visible around $x = 5$. This could be cause of the small slope of $f(x)$ around $x = -1$ and $x = 5$ (Figure 3). In Equation 2 the term $f'(x)$ is in the denominator, and small values of $f'(x)$ may cause overflow, or result in very large values of x_{x+1} that are far away from the true root.

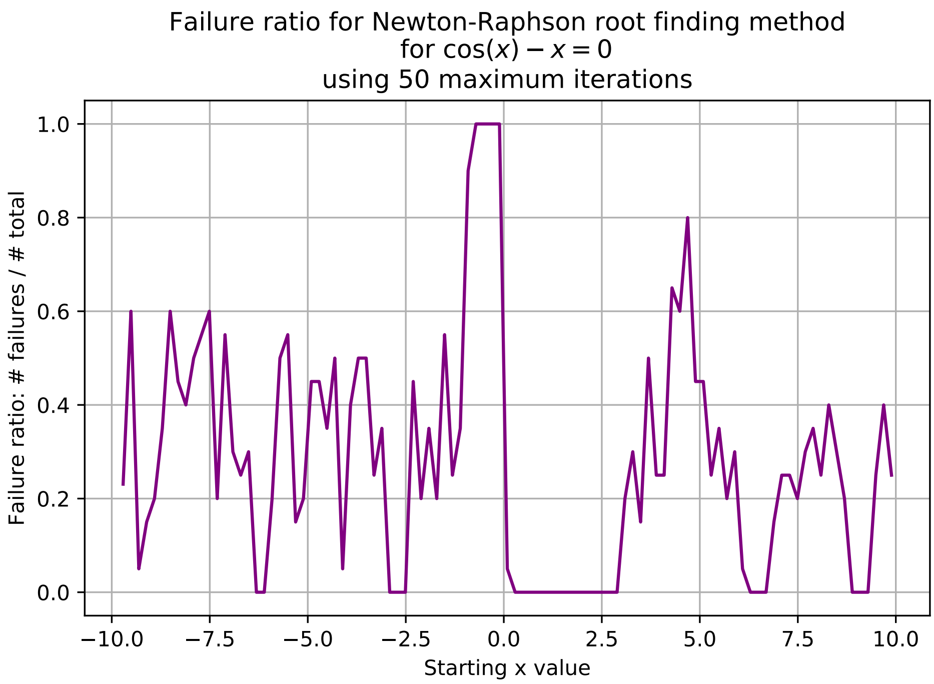

Figure 2: Failure ratio of Newton-Raphson root finding method for 50 maximum number of iterations. Failure ratio is close to zero in the interval $[0, 2.5]$

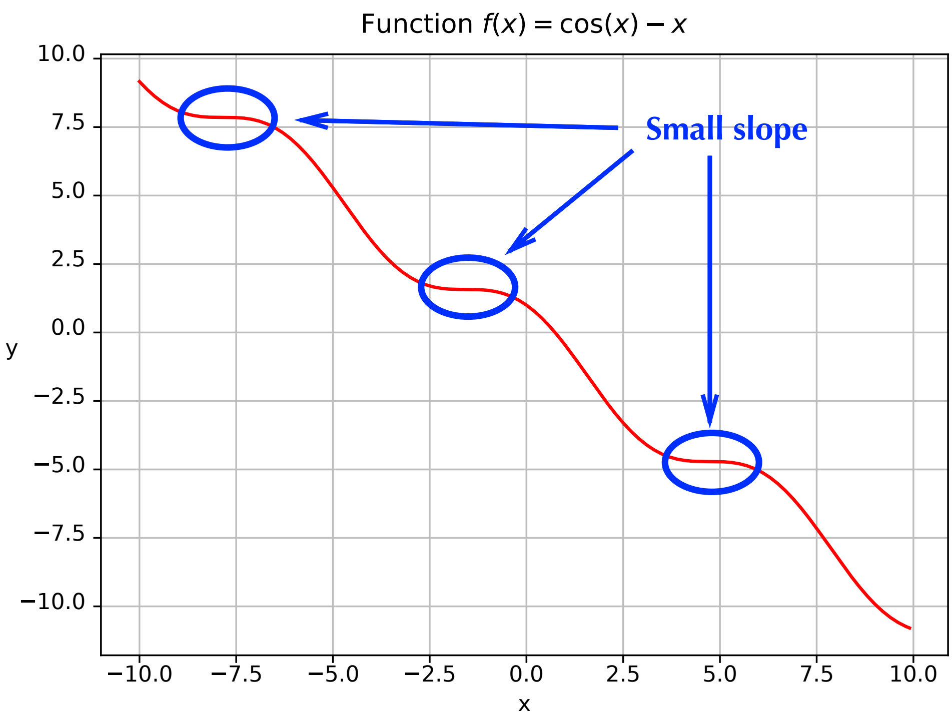

Figure 3: Function $f(x) = \cos(x) - x$.

We have found that the failure ratio decreased as we increased N_{max}. On Figure 4 we can see the failure rate below 0.1 for most values of x_0 when we used very large number of maximum iterations (N_{max} = 100 000). Interestingly, almost all calculations were still failing for $x_0 \in [-1, 0]$, while values of x_0 from the other two regions of small slope of $f(x)$ (for $x_0 \approx -7.5$ and $x_0 \approx 5$) had very low failure rates below 0.1. It is unclear why we saw such a difference in failure rates for regions of $f(x)$ that have same rates of change.

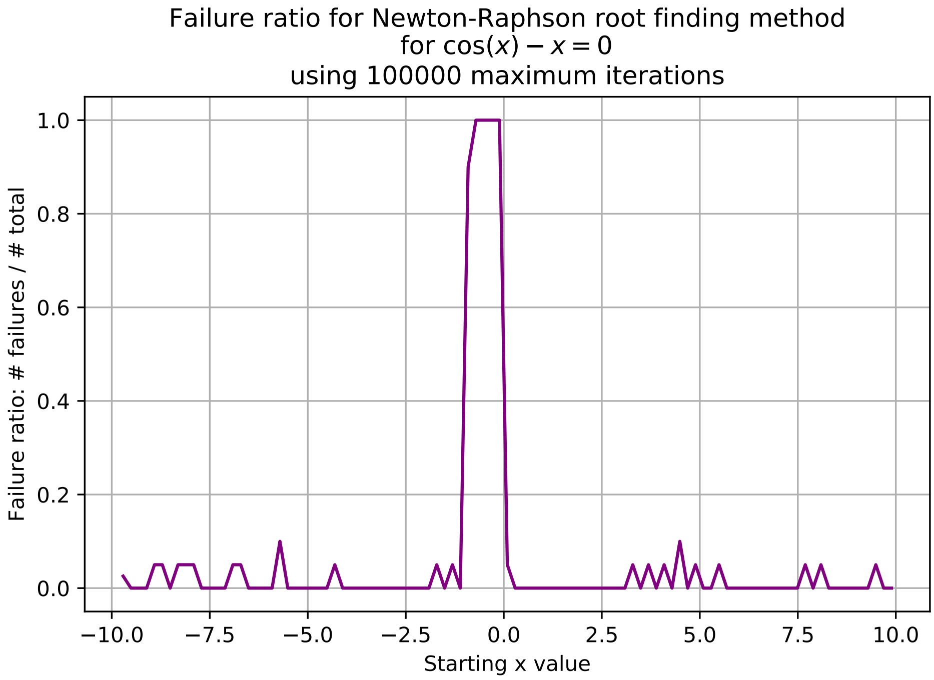

Figure 4: Failure ratio of Newton-Raphson root finding method for 100 000 maximum number of iterations. Failure ratio is close to zero in the interval $[0, 2.5]$

Failure ratio dependence on N_{max}

Here we investigate how the failure ratio depends on the maximum number of iterations (Figure 5). We can see that the failure ratio is lower with higher N_{max}. This is not an unexpected result. When we allow the program to run the calculation from Equation 2 more times, it might make it more likely for the program to converge.

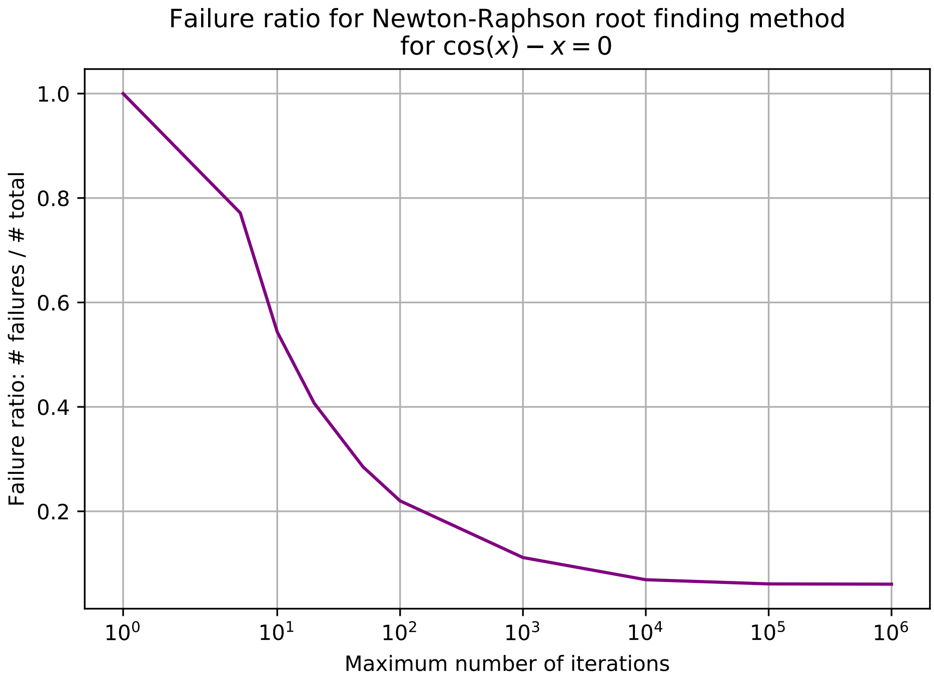

Figure 5: Failure ratio of Newton-Raphson root finding method different values of maximum number of iterations.

Result

We ran the program with initial value $x_0 = 0.5$, maximum number of iterations $N_{max} = 20$ and tolerance $\epsilon = 1 \times 10^{-5}$. The program approximated the root to be 0.73909. Our result agrees with the root calculated by Mathematica code shown in Listing 2.

Listing 2: Finding root of $\cos(x) - x = 0$ using Mathematica.

```
FindRoot[Cos[x] - x, {x, 0.5}]
{x -> 0.739085}
```